RUBY ITERATORS - EACH AND COLLECT

Iterators are nothing but methods supported by *collections*. Objects that store a group of data members are called collections. In Ruby, arrays and hashes can be termed collections.

Iterators return all the elements of a collection, one after the other. We will be discussing two iterators here, *each* and *collect*. Let's look at these in detail.

Ruby *each* Iterator:

The each iterator returns all the elements of an array or a hash.

Syntax:

```
collection.each do |variable|
   code
end
```

Executes *code* for each element in *collection*. Here *collection* could be an array or a ruby hash.

Example:

```
#!/usr/bin/ruby

ary = [1,2,3,4,5]
ary.each do |i|
   puts i
end
```

This will produce following result:

```
1
2
3
4
5
```

You always associate the *each* iterator with a block. It returns each value of the array, one by one, to the block. The value is stored in the variable **i** and then displayed on the screen.

Ruby *collect* Iterator:

The *collect* iterator returns all the elements of a collection.

Syntax:

```
collection = collection.collect
```

The *collect* method need not always be associated with a block. The *collect* method returns the entire collection, regardless of whether it is an array or a hash.

Example:

```
#!/usr/bin/ruby
```

```
a = [1,2,3,4,5]
b = Array.new
b = a.collect
puts b
```

This will produce following result:

```
1
2
3
4
5
```

NOTE: The *collect* method is not the right way to do copying between arrays. There is another method called a *clone* which should be used to copy one array into another array.

You normally use the collect method when you want to do something with each of the values to get the new array. For example, this code produces an array b containing 10 times each value in a.

```
#!/usr/bin/ruby

a = [1,2,3,4,5]
b = a.collect{|x| 10*x}
puts b
```

This will produce following result:

```
10
20
30
40
50
```